УДК 004.41

Е. А. Юлюгин

Intel Corporation

Реализация технологии прямого исполнения обращений к счетчику TSC в программном симуляторе

Исследование проводилось с целью увеличения производительности сценариев программного моделирования, демонстрирующих частые обращения с счетчику TSC (Time Stamp Counter) при исполнении на процессорах с архитектурой Intel® 64. Для достижения поставленной цели был разработан алгоритм, разрешающий прямое исполнение инструкций чтений счетчика TSC. Предложенный алгоритм был реализован и протестирован в полноплатформенном программном симуляторе Wind River® Simics®.

Ключевые слова: Simics, TSC, VT-x, моделирование, гипервизор, прямое исполнение.

E. A. Yulyugin

Intel Corporation

Implementation of direct time stamp counter access execution technique in a software simulator

The goal of this research is to improve the performance of software simulation workloads frequently accessing TSC (Time Stamp Counter), while running on Intel® 64 platforms. To achieve the goal a new direct TSC access execution approach is developed, implemented and tested by the full-platform software simulator Wind River® Simics®.

Key words: Simics, TSC, VT-x, simulation, hypervisor, direct execution.

1. Введение

Моделирование является неотъемлемой частью процесса разработки современных вычислительных систем. С помощью программных моделей можно выявить ошибки проектирования, разработать и отладить программное обеспечение до производства реальной системы, тем самым существенно сократив время, необходимое для выхода конечного продукта на рынок.

Программная модель должна демонстрировать высокую скорость работы для того, чтобы быть применимой для отладки сложных сценариев, таких как загрузка операционных систем и гипервизоров. В случае, когда архитектура симулируемой системы совпадает с хозяйской, максимальной производительности можно добиться за счёт прямого исполнения гостевого кода на хозяйской аппаратуре. Поэтому большинство современных виртуальных машин [1–4], моделирующих архитектуру Intel® 64, используют технологию Intel VT-x [5] в качестве инструмента, обеспечивающего поддержку прямого исполнения.

Предпосылками данного исследования послужили сценарии программного моделирования, демонстрирующие недостаточно высокую скорость работы при частых обращениях к счетчику TSC (Time Stamp Counter), во время исполнения под управлением функционального симулятора Wind River® Simics® [2]. К таким сценариям относится загрузка гипервизоров, таких как VMware® ESXi [1], и виртуальных машин под их управлением.

Целью данного исследования является изучение причин недостаточно быстрой работы Simics при моделировании проблемных сценариев, а также разработка методов, позволяющих увеличить их производительность.

2. Обзор симулятора Wind River Simics

Wind River Simics — функционально точный полноплатформенный симулятор, который предоставляет возможность создавать высокопроизводительные модели цифровых вычислительных устройств. Разработанные с помощью Simics модели позволяют запускать, исследовать, тестировать, а также заниматься отладкой немодифицированного программного обеспечения. Simics создает виртуальное окружение, способное работать как с низкоуровневым ПО, таким как BIOS и операционные системы, так и с любыми приложениями, работающими на уровне пользователя.

Для обеспечения высокой скорости моделирования Simics поддерживает несколько режимов симуляции центрального процессора. Переключения между режимами происходят динамически в зависимости от исполняемого гостевыми приложениями кода. Для моделирования процессоров, основанных на архитектуре Intel 64, в Simics предусмотрены следующие режимы исполнения гостевого кода.

Интерпретация — наиболее универсальный, однако самый медленный режим симуляции. Применяется для моделирования функционально сложных, а также редко встречающихся инструкций.

Двоичная трансляция применяется для моделирования простых и наиболее часто встречающихся инструкций. Данный режим значительно быстрее интерпретации, однако является более сложным в разработке и поддержке.

VMP — режим прямого исполнения, основанный на технологии аппаратной виртуализации Intel VT-x [5]. Данный режим дает возможность добиться максимальной скорости работы, однако не позволяет моделировать все инструкции. Команды, отсутствующие в хозяйской аппаратуре, а также служебные и привилегированные инструкции требуют программной симуляции с помощью интерпретации или двоичной трансляции.

Более подробное описание каждой из технологий можно найти в [7]. Данная работа посвящена изучению режима прямого исполнения как наиболее актуального и быстрого метода моделирования архитектуры Intel 64.

Современные вычислительные комплексы могут содержать несколько процессоров, а также периферийные устройства, принципы работы которых значительно отличаются. В реальности они работают одновременно, чего при моделировании добиться невозможно.

Большинство существующих симуляторов, включая Simics, используют модель дискретных событий (*англ.* Discrete Event Simulation, DES) [6,7] для симуляции полной платформы, состоящей из множества процессоров и периферийных устройств. В этой модели событие — изменение какого-либо состояния устройства. События происходят мгновенно и только в определенные моменты виртуального времени. Если на одно и то же время запланировано более одного события, то они обычно обрабатываются в порядке добавления в очередь. Каждое событие может порождать новые в будущем или отменять уже запланированные. Создание событий в прошлом запрещено. Продвижение симулируемого времени при этом происходит интервалами, равными расстоянию между ближайшими событиями.

Следует отметить, что модель дискретных событий не является оптимальной с точки зрения производительности для симуляции центрального процессора, так как его состояние изменяется на каждом такте. По этой причине при моделировании полной платформы используется подход, комбинирующий DES и симуляцию исполняющих устройств (рис. 1).

Последовательность действий при моделировании полной платформы с применением подхода, комбинирующего модель дискретных событий и симуляцию исполняющих устройств:

1) вычисление времени до следующего необработанного события в очереди,

2) передача управления в модель процессора на длительность, не превышающую вычисленную на первом шаге,

3) продвижение времени в моделируемой системы в соответствии с числом тактов, фактически исполненных процессором,

4) обработка достигнутых событий, если необходимо,

5) повторение цикла.

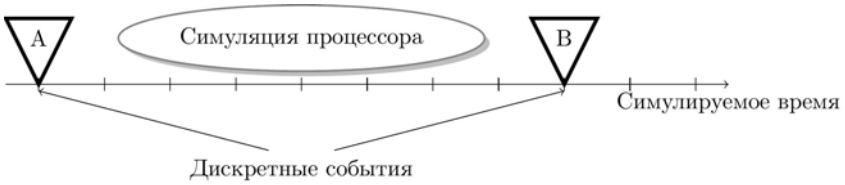


Рис. 1. Комбинация модели дискретных событий и симуляции процессора

3. Моделирование обращений к счетчику TSC

За время развития вычислительных систем было создано множество устройств измерения отрезков времени. Наиболее широкое распространение в текущий момент получил счетчик TSC [5]. Данный счетчик при исполнении на архитектуре Intel 64 активно используют современные операционные системы, такие как Linux и Microsoft Windows, а также многие гипервизоры и пользовательские приложения.

Текущее значение счетчика хранится в 64-битном регистре и может быть получено с помощью трех инструкций: RDTSC, RDMSR и RDTSCP. Инструкция RDMSR доступна только в режиме супервизора, тогда как исполнение команд RDTSC и RDTSCP может быть разрешено и в режиме пользователя. Инструкция RDTSCP позволяет получить идентификатор процессора одновременно со значением счетчика, что может быть использовано для проверки того, что последовательные считывания счетчика произошли на одном и том же вычислительном ядре.

Существующие подходы к обработке доступов к TSC при прямом исполнении можно разделить на три типа.

Программное моделирование — наиболее простое и точное решение, которое поддерживается большинством гипервизоров, включая VMware ESXi, Simics, KVM [3] и Xen [4]. Данный подход также является самым медленным, так как приводит к накладным расходам, вызванным переключением между режимам программного моделирования и прямого исполнения. Следует отметить, что KVM является модулем операционной системы Linux и осуществляет программное моделирование в режиме супервизора, тем самым сокращая расходы на переключение режимов работы процессора.

Прямое исполнение обращений к счетчику TSC позволяет исключить накладные расходы, вызванные переключениями режимов моделирования при его чтении, однако делает поведение модели недетерминированным из-за особенностей архитектуры Intel 64. Также данное решение является более сложным с точки зрения реализации и поддержки. Поддерживается в KVM и VMware ESXi.

Паравиртуализация позволяет достичь скорость, близкую к прямому исполнению, сохраняя детерминированность и точность программного моделирования. Однако данное решение требует модификации гостевого программного обеспечения, что зачастую невозможно. Поддерживается в Xen.

4. Технология аппаратной виртуализации Intel VT-x

Основой эффективной виртуализации процессоров архитектуры Intel 64 является технология Intel VT-x [5], расширяющая возможности процессора новыми инструкциями и

режимами работы. Монитор виртуальных машин исполняется в режиме VMX root, в котором доступны команды, позволяющие контролировать поведение виртуальных машин, исполняющихся в режиме VMX non-root. В режиме VMX non-root поведение процессора изменено так, что определенные инструкции и события вызывают VM exit — передачу управления в монитор виртуальных машин, который после выполнения действия, соответствующего причине выхода, может вернуться к исполнению кода виртуальной машины, снова перейдя в режим VMX non-root.

В режиме VMX non-root поведение инструкций, считывающих значение счетчика TSC, зависит от конфигурации структуры VMCS [5] следующим образом: если ее поле «RDTSC exiting» имеет значение 0, то инструкции RDTSC и RDTSCP не вызывают VM exit. Поведение инструкции RDTSCP дополнительно контролируется значением поля «enable RDTSCP». Возвращаемое значение будет равно текущему значению счетчика, просуммированному с величиной «TSC offset», если выставлен флаг «Use TSC offsetting».

В расширении Intel VT-x также предусмотрен вытесняющий таймер (*англ.* VMX-Preemption Timer) [5], который может быть использован для ограничения времени, проведенного в режиме прямого исполнения.

5. Описание предложенного алгоритма

Планирование исполнения происходит в режиме пользователя, когда доступна очередь событий, являющаяся частью программной модели. Из очереди событий вычисляется количество циклов ΔT, необходимое для достижения ближайшего события. Прямой доступ к счетчику TSC допускается, если ΔT превышает заданный порог, обозначаемый как *rdtsc_threshold*. В противном случае прямое исполнение также может быть разрешено, однако обращения к TSC будут вызывать VM exit. Далее происходит переход в режим супервизора, в котором исполняется монитор виртуальных машин, отвечающий за прямое исполнение (рис. 2).

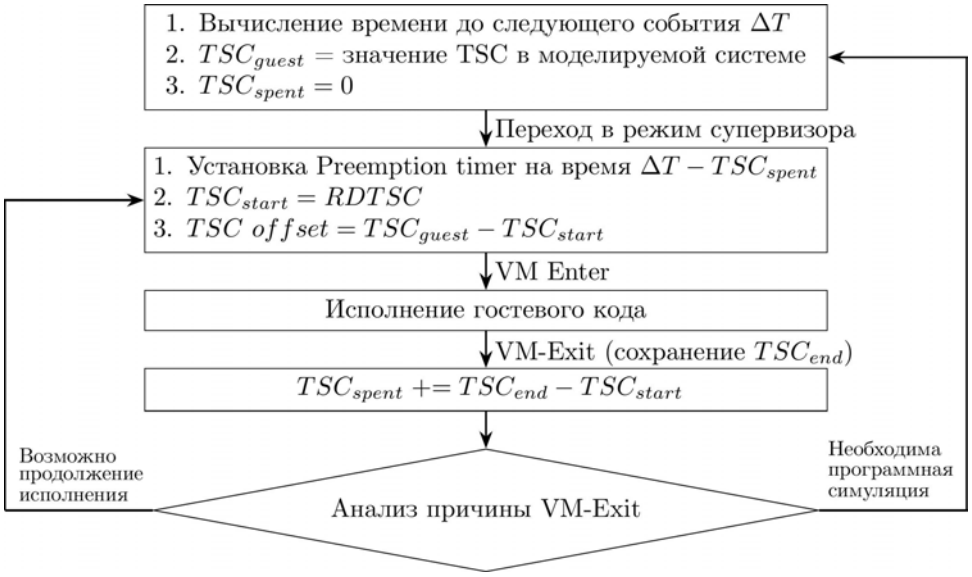


Рис. 2. Последовательность действий при прямом исполнении доступов к счетчику TSC

Если режим прямого обращения к счетчику TSC включен, то монитор виртуальных машин настраивает структуру VMCS так, чтобы разрешить исполнение инструкций чтения TSC, как описано в секции 4. После этого происходит установка значения Preemption timer в соответствии с полученным ранее ΔT, вычисление разницы значений счетчика в моделируемой и хозяйской системах и обновление поля TSC offset. Затем осуществляется переход в режим VMX non-root, в котором исполняется код моделируемой системы до тех пор, пока не произойдет VM Exit. Структура VMCS должна быть настроена так, чтобы текущее зна-

чение счетчика TSC было сохранено при переходе в режим VMX root. Через полученное значение определяется величина TSC_{spent} — количество циклов, проведенных в прямом исполнении. В зависимости от причины выхода монитор виртуальных машин может либо возобновить прямое исполнение, либо передать управление программной модели.

После выхода из режима супервизора значение времени в моделируемой системе обновляется в соответствии с TSC_{spent}. Следует отметить, что описанным выше способом вычисления TSC_{spent} можно получить только верхнюю оценку количества циклов, проведенных в режиме VMX non-root. Более точной оценки можно достичь, используя значения, полученные через Preemption timer, как это предлагается в работе [8]. Однако использование разности значений Preemption timer для продвижения моделируемого времени не является корректным, так как не обеспечивает монотонность времени в моделируемой системе.

При продвижении моделируемого времени важно не только учесть количество циклов, проведенных в режиме прямого исполнения, но и гарантировать, что значения TSC, полученные после выхода из режима прямого исполнения, будут превышать значения счетчика, полученные в режиме VMX non-root. Это условие может быть записано в виде (5.1):

$$RDTSC_{guest} - RDTSC_{start} \leq Ptimer_{stop} - Ptimer_{start}, \tag{5.1}$$

где $RDTSC_{guest}$ — результат исполнения инструкция RDTSC в режиме прямого исполнения; $RDTSC_{start}$ — значение счетчика, используемое для вычисления смещения, устанавливаемого в поле «TSC offset»; $Ptimer_{start}$ и $Ptimer_{stop}$ — значения TSC в моменты запуска и остановки Preemption timer. Последовательность этих событий изображена на рис. 3.

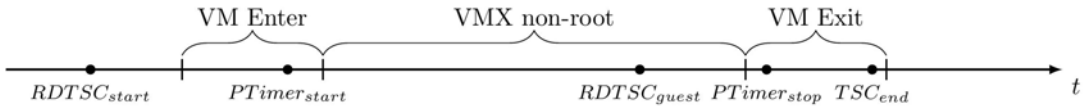


Рис. 3. Временная диаграмма событий, происходящих при запуске прямого исполнения с разрешенными обращения к счетчику TSC

Оценить или ограничить время между событиями $RDTSC_{start}$ и $Ptimer_{start}$ невозможно, так как в этот промежуток времени может, например, произойти переход в режим системного управления (*англ.* System Management Mode). Таким образом, невозможно гарантировать выполнение условия (5.1), которое может быть представлено в виде (5.2):

$$Ptimer_{start} - RDTSC_{start} \leq Ptimer_{stop} - RDTSC_{guest} \tag{5.2}$$

Наличие $rdtsc\_threshold$ связано с тем, что каждый запуск прямого исполнения с разрешенными обращениями к счетчику TSC привносит ошибку, вызванную временем, необходимым на переходы между режимами работы процессора и не фиксированным временем между моментом считывания хозяйского счетчика $RDTSC_{start}$ и началом выполнения процедуры VM Enter. Требование $\Delta T > rdtsc\_threshold$ позволяет уменьшить количество случаев, когда ошибка, внесенная переходами, будет превышать полезную работу, связанную с непосредственным исполнением кода моделируемой системы. Изначально параметр $rdtsc\_threshold$ имел значение 10.000, равное удвоенному количеству циклов, которое в среднем необходимо для того, чтобы запустить и завершить режим прямого исполнения с разрешенными обращениями к TSC.

6. Измерения

Эксперименты проводились на рабочей станции с одним центральным процессором Intel Core™ i7-6700K 4 Ггц и 8 Гбайт ОЗУ, использовалась 64-битная операционная система SUSE Linux Enterprise Server 11. Измерения проводились на симуляторе Wind River Simics версии 5.

Для всех исследуемых сценариев изучалось значение ускорения, полученного за счет прямого исполнения обращений к счетчику TSC. Данная величина определялась как отношение времени работы симулятора при программном моделировании TSC к времени, полученному в режиме с прямым исполнением обращений к счетчику.

Тестирование проводилось в несколько этапов. Сначала на синтетическом тесте было определено максимальное ускорение, которое можно получить за счет прямого исполнения доступов к счетчику TSC. Далее было проведено тестирование производительности сценариев работы гипервизоров. В конце была проверена производительность на общем наборе тестов исследуемой платформы.

6.1. Определение максимально возможного ускорения

Для оценки максимального ускорения был написан тест, исполняющий инструкцию RDTSC 100 раз в цикле, состоящем из 10.000.000 итераций (листинг 21.1).

Листинг 21.1. Программа для сравнения скорости моделирования инструкции RDTSC

```
void _start() {
    register uint32_t hi, lo;

    magic_instruction();
    for (int i = 0; i < RDTSC_NUM; i++) {
        __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi)); // (0)
        ...
        __asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi)); // (99)
    }
    magic_instruction();
}
```

При разрешенном прямом исполнении RDTSC выполнение теста заняло 7 секунд, при запрещенном — 425 секунд. Максимально возможный прирост производительности, полученный за счет прямого исполнения команды RDTSC, равен 61.

6.2. Тестирование производительности гипервизоров

Результаты тестирования производительности моделирования загрузки различных гипервизоров приведены на рис. 4.

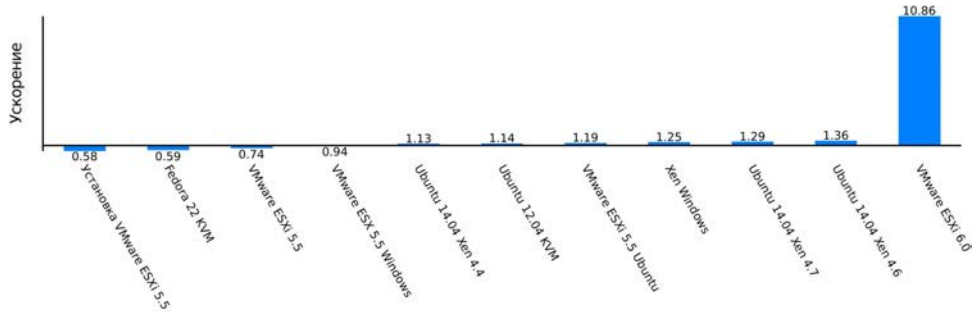


Рис. 4. Результаты тестирования производительности сценариев загрузки гипервизоров

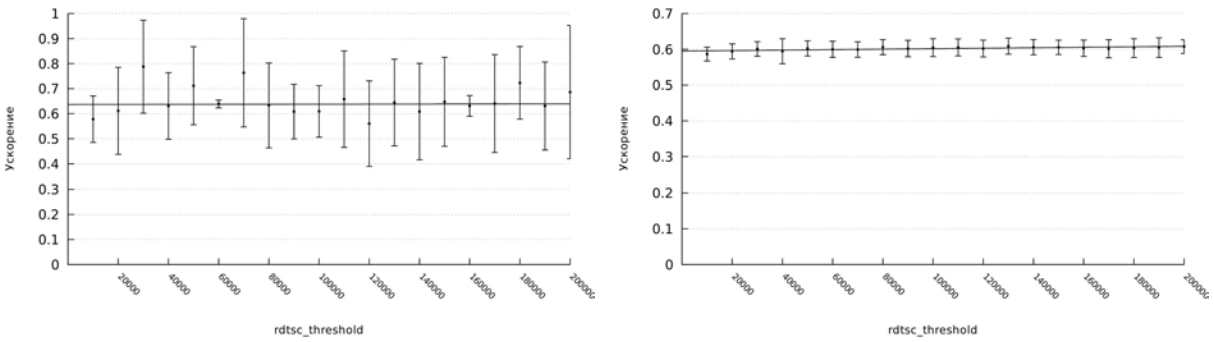
Среднее геометрическое улучшение производительности — 1,22. Наилучший результат — прямое исполнение RDTSC — показывает при загрузке гипервизора VMware ESXi 6.0 без клиента, где по причине RDTSC происходило 94% всех выходов из режима прямого исполнения. Наибольшая потеря производительности наблюдается при моделировании сценария установки гипервизора VMware ESXi 5.5. Негативный результат объясняется конфликтом с другой оптимизацией — rdtsc_slow_cycles. Во время установки гипервизора VMware ESXi 5.5 наблюдается большое количество циклов активного ожидания (*англ.* busy-wait loop). Параметр rdtsc_slow_cycles задает количество моделируемых циклов, необходимое на выполнение инструкций чтения счетчика TSC. По умолчанию данный параметр имеет значение 20.000, которое на порядки превышает латентность этой инструкции

в реальной системе, тем самым позволяя значительно снизить количество итераций цикла ожидания и увеличивая скорость моделирования подобных участков. Применение такой оптимизации при прямом исполнении доступов к счетчику невозможно. Ускорение сценария установки VMware ESXi 5.5 равняется 1,03 при отключенной оптимизации циклов ожидания (rdtsc_slow_cycles = 0).

Замедление остальных тестов объясняется взаимодействием с технологией, направленной на уменьшение количества выходов из режима прямого исполнения [9]. Данная технология на основе анализа истории событий VM exit делает предположение о необходимости перехода в режим прямого исполнения, тем самым уменьшая количество случаев, когда накладные расходы, связанные с частыми переключениями режимов работы симулятора, превышают полезную работу — исполнение инструкций моделируемой системы.

Анализ причин выходов из режима прямого исполнения показал, что при включении прямого исполнения доступов к счетчику TSC увеличивается количество исполненных команд для синхронизации процессоров — PAUSE. Что объясняется рассинхронизацией значений TSC, получаемых при прямом исполнении за счет ошибки, вызванной невозможностью точно учесть количество циклов, проведенных в режиме VMX non-root. Исследование сценария загрузки гипервизора VMware ESXi 5.5 показало 58-кратное увеличение числа событий VM exit, вызванных инструкцией PAUSE.

Измерения показали отсутствие зависимости между значением параметра rdtsc_threshold и скоростью работы симулятора на исследуемых сценариях. Результаты измерений для сценариев установки VMware ESXi 5.5 и загрузки операционной системы Fedora 22 с работающим модулем KVM приведены на рис. 5.



а) Установка VMware ESXi 5.5 б) Загрузка ОС Fedora 22 с KVM

Рис. 5. Зависимость ускорения от значений параметра rdtsc_threshold

Однако в целом исследуемые сценарии продемонстрировали более стабильную работу в случае, когда параметр rdtsc_threshold равнялся 50.000. Данное значение было выбрано в качестве значения по умолчанию. Среднее геометрическое улучшение производительности моделирования гипервизоров при выбранном значении параметра получилось равным 1,36.

6.3. Общий набор тестов

У исследуемой платформы существует широкий набор тестов, включающий загрузку различных операционных систем и гипервизоров. Из этого набора были выбраны сценарии, продолжительность моделирования которых составляет более 50 секунд. Полученная выборка тестов использовалась для изучения ускорения, получаемого за счет прямого исполнения обращений к счетчику TSC. Результаты приведены на рис. 6.

Среднее геометрическое ускорение, вычисленное для тестов неиспользующих виртуализацию, равняется 0,99. То есть производительность нецелевых сценариев осталась неизменной. Среднее геометрическое значение ускорение на полном наборе тестов, включающем как тесты гипервизоров, так и все остальные сценарии, равняется 1,12. Таким образом, разработанный алгоритм позволил увеличить скорость работы программного симулятора на

полном наборе тестов исследуемой платформы. Следует отметить, что предложенная оптимизация позволила сократить время работы не только на целевых сценариях, но и при моделировании загрузки некоторых операционных систем, таких как Microsoft Windows server 2016.

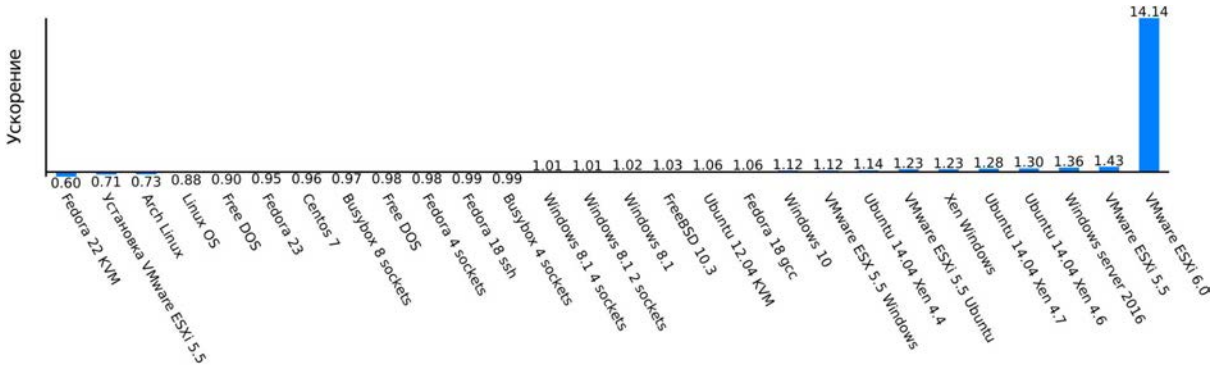


Рис. 6. Результаты измерений производительности полного набора тестов исследуемой платформы

7. Заключение

В данной статье описаны модификации полноплатформенного функционально точного симулятора Wind River Simics, расширяющие существующий в Simics механизм прямого исполнения инструкций архитектуры Intel 64 возможностью моделирования команд чтения счетчика TSC без переходов в режим программной виртуализации.

Было проведено всестороннее тестирование разработанного метода, включающее в себя узконаправленные модульные тесты, сценарии загрузки гипервизоров, являющиеся целевыми, а также общий набор тестов исследуемой платформы. Улучшение производительности было достигнуто как на целевых сценариях, так и на некоторых тестах, моделирующих загрузку операционных систем.

Негативными эффектами предложенного метода являются потеря детерминированности и рассинхронизация значений счетчика TSC при моделировании многоядерной системы. Последний эффект приводит к увеличению количества выходов из режима прямого исполнения, вызванных инструкцией PAUSE, что может негативно сказаться на производительности симулятора.

Дальнейшая работа по улучшению описанного алгоритма должна состоять в разработке метода, позволяющего динамически определять циклы активного ожидания и отключать прямое исполнение обращений к счетчику TSC при моделирования этих участков. Также необходимо внести изменения в планировщик Simics, разработав более эффективную схему распределения ресурсов между моделируемыми процессорами.

Литература

1. Information Guide: Timekeeping in VMware Virtual Machines. VMware, 2008.
2. *Aarno D., Engblom J.* Software and System Development using Virtual Platforms: Full System Simulation with Wind River Simics. Morgan Kaufmann Publishers, 2014.
3. *Amsden Z.* Timekeeping Virtualization for X86-Based Architectures. 2010.
4. *Magenheimer D.* TSC_MODE HOW-TO.
URL: http://xenbits.xen.org/docs/4.3-testing/misc/tscmode.txt.
5. Intel® 64 and IA-32 Architectures Software Developer's Manual. — Intel Corporation, 2017.
6. *Albrecht M.C.* Introduction to discrete event simulation. 2010.

7. *Речистов Г.С., Юлюгин Е.А., Иванов А.А., Шишпор П.А., Щелкунов Н.Н., Гаврилов Д.А.* Основы программного моделирования ЭВМ. 2-е издание. Долгопрудный: МФТИ, 2013.

8. *Конычев В.В., Юлюгин Е.А., Речистов Г.С.* Оптимизация сценариев программного моделирования, использующих команду чтения времени RDTSC. Программа 58-й научной конференции МФТИ. 2015. С. 14.

9. *Кравцов А.А., Речистов Г.С., Юлюгин Е.А.* Увеличение производительности режима прямого исполнения в программном симуляторе. Программа 58-й научной конференции МФТИ. 2015. С. 14.

References

1. Information Guide: Timekeeping in VMware Virtual Machines. VMware, 2008.

2. *Aarno D., Engblom J.* Software and System Development using Virtual Platforms: Full System Simulation with Wind River Simics. Morgan Kaufmann Publishers, 2014.

3. *Amsden Z.* Timekeeping Virtualization for X86-Based Architectures. 2010.

4. *Magenheimer D.* TSC_MODE HOW-TO.
URL: http://xenbits.xen.org/docs/4.3-testing/misc/tscmode.txt.

5. Intel® 64 and IA-32 Architectures Software Developer's Manual. Intel Corporation, 2017.

6. *Albrecht M.C.* Introduction to discrete event simulation. 2010.

7. *Rechistov G.S., Yulyugin E.A., Ivanov A.A., Shishpor P.A., Shchelkunov N.N., Gavrilov D.A.* Foundations of Software Simulation, 2nd edition. Dolgoprudny:MIPT, 2013.

8. *Konychev V.V., Yulyugin E.A., Rechistov G.S.* Optimization of software simmulation workloads using RDTSC instruction. Program of 58th MIPT scientific conference. 2015. P. 14.

9. *Kravtsov A.A., Rechistov G.S., Yulyugin E.A.* Performance improvements of direct execution mode in the software simulator. Program of 58th MIPT scientific conference. 2015. P. 14.

Поступила в редакцию 15.11.2017